NASA-TM-112265

TM-82-TM
015770

Experience with Parallel Computers at NASA Ames

David H. Bailey

RNR Technical Report RNR-91-007
February 7, 1991

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035

ARC 275 (Rev Feb 81)

Experience with Parallel Computers at NASA Ames

David H. Bailey

RNR Technical Report RNR-91-007
February 7, 1991

Abstract

Beginning in 1988, the Numerical Aerodynamic Simulation (NAS) organization at NASA Ames Research Center has studied the usability of highly parallel computers on computational fluid dynamics and other aerophysics applications. Presently the NAS Applied Research Branch operates a CM-2 with 32,768 nodes and an Intel iPSC/860 with 128 nodes. This note gives an overview of our experience in using these systems, highlights both strong points and weak points of each, and discusses what improvements will be required in future highly parallel systems in order that they can gain acceptance in the mainstream of scientific computing.

The author is with the NAS Applied Research Branch at NASA Ames Research Center, Moffett Field, CA 94035.

Introduction

NASA Ames Research Center has long been a trail blazer in the field of high performance scientific computation, particularly for computational fluid dynamics and other aerophysics applications. Back in the 1970s NASA Ames was the home of the Illiac IV, which featured a peak performance of approximately 250 MFLOPS. In 1982, NASA Ames took delivery of the first two processor Cray X-MP, which featured a peak performance of nearly 400 MFLOPS. In 1986 the newly formed Numerical Aerodynamic Simulation (NAS) organization at NASA Ames installed the first full-memory Cray-2, with a peak performance of 1.6 GFLOPS. Most recently, the NAS facility added in 1988 a Cray Y-MP with a peak performance of 2.6 GFLOPS.

By the late 1980s it became clear that the "classical" supercomputer design epitomized by existing Cray systems was approaching maturity, and that highly parallel computer systems had the most potential for achieving multi-TFLOPS levels of sustained performance, which we project will be required before the end of the decade. In order to better understand these systems and to help advance this technology to the level required for mainstream supercomputing, the NAS Applied Research Branch was organized in 1988. Scientists in this organization and its affiliates port application programs to parallel computers, develop new algorithms and programming techniques, and study the performance of the resulting implementations [1, 2, 3, 5, 6, 7, 8 and 9]. We now operate a Connection Machine 2 (CM-2) by Thinking Machines, Inc. and an iPSC/860 by Intel Scientific Computers. We plan to acquire significantly more powerful systems in the future.

As of this date, we have been using the CM-2 for nearly three years and the iPSC/860 for over one year. This note gives a brief summary of our experience and highlights both the strengths and the weaknesses that we have observed in these two systems. Requirements and recommendations for future systems are then outlined.

The Connection Machine

The NAS CM-2, obtained in collaboration with the Defense Advanced Projects Research Association (DARPA), was installed in 1988 with 32,768 processors. Recently it was upgraded with 64 bit floating point hardware and four gigabytes of main memory, or about twice that of the NAS Cray-2. We hope soon to install the new "slicewise" Fortran compiler, which will view the hardware as a SIMD array of 64 bit floating point processing units.

For the first year or two, most applications on the CM-2 were coded in *LISP, an extension of the LISP language for parallel processing. More recently, however, programmers have started using the CM Fortran language, which is based on Fortran-90 [4]. CM Fortran has been plagued by delays, and it is still in a "pre-release" stage.

Perhaps the most notable achievement on the CM-2 is the porting of a variation of the NASA Ames "ARC3D" code. This is a three dimensional implicit Navier-Stokes fluid dynamics application and employs a number of state of the art techniques. On a $64 \times 64 \times 32$ grid, this code now runs at 190 MFLOPS on 16,384 processors [7]. The equivalent one processor Y-MP rate for this code is 128 MFLOPS.

The strongest positive feature of the CM-2, from our perspective, is TMC's adoption of a programming language based on Fortran-90. Some have questioned whether the Fortran-90 array constructs are sufficiently powerful to encompass a large fraction of modern scientific computation. Basically, this is the same as the question of to what extent the single program multiple data (SPMD) programming model encompasses modern scientific computation. While we cannot speak for other facilities, it seems clear that a very large fraction of our applications can be coded effectively using this model. Even some applications, such as unstructured grid computations, which appeared at first to be inappropriate for SPMD computation, have subsequently been successfully ported to the CM-2, although different data structures and implementations techniques have been required [5].

One weak point of the current CM-2 is that it utilizes a rather dated hardware technology. Further, the CM was not designed from the beginning for floating point computation, and the floating point processors now in the system were incorporated in a somewhat inelegant fashion. Clearly future editions of the CM need to employ more advanced technology and to integrate floating point processing more effectively. Secondly, we have found the fact that the system cannot be partitioned any finer than into 8,192 node blocks to be disadvantageous in a multiuser setting, particularly for users debugging and upgrading their codes. One additional weak spot is the slowness of the router. It has been our experience that if a program must make significant use of the router, its performance on 8,192 nodes is often no better than on a state of the art RISC workstation.

The principal weakness with the CM-2 at the present time, however, is the CM Fortran compiler, which still has a number of bugs and frequently delivers disappointing performance. Partly this is due to the fact that writing an effective Fortran-90 compiler, including efficient array intrinsics, has proved to be much more of a challenge than originally anticipated. Other vendors attempting to write Fortran-90 compilers are reported to be having similar difficulties. In any event, clearly the CM Fortran compiler must be further improved.

Another major weakness of the CM-2 is the extent to which the achieved performance is sensitive to the algorithm selected. It is not unusual for a reasonably well conceived and well written code to deliver single digit MFLOPS performance rates. Often the programmer must try a number of different parallel algorithms and implementation schemes before finding one that delivers respectable performance. Some scientists have lost interest in the system after such experiences. This phenomenon is partly due to the fact that a very high level of parallelism is required to obtain good performance on the CM-2. But it is also due in part to various weaknesses in the compiler.

The Intel iPSC/860

The NAS Intel iPSC/860 system, which was one of the first two of these systems shipped by Intel, was installed in January 1990. It has 128 nodes, each of which contains a 40 MHz i860 processor (with 60 MFLOPS peak performance) and 8 megabytes of memory. The complete system has one gigabyte of memory and a theoretical peak performance of roughly seven GFLOPS.

Programs for the Intel system may be coded in either C or Fortran, although most scientists have chosen Fortran. At first, the only available compilers were from Green Hills. These compilers did not utilize many of the advanced features of the i860, and so single node performance was predictably poor, typically only one or two MFLOPS. More importantly, compile/link times were very long, typically 30 minutes for a 5,000 line program. Recently Intel made available some new Fortran and C compilers, produced by the Portland Group. While the initial release of these compilers did not feature significantly faster execution speeds, at least the compile/link speeds were much better. Further, the Portland Group's i860 compilers are now available on Sun-4 workstations, and running them off-line on Sun-4 systems has reduced compile/link times by a factor of ten, to only about seventy seconds for a 5,000 line program.

Typical of the performance results obtained on the Intel system so far is a performance rate of 123 MFLOPS using 64 nodes on the NASA Ames "ARC2D" code, a two dimensional fluid dynamics program with a 320×128 grid [2, 3]. This compares with 86 MFLOPS on a single processor of the Cray-2 and 172 MFLOPS on a single processor of the Cray Y-MP.

An important positive feature of the Intel system is the fact that programs can be easily ported to a single node, which is not unlike the RISC workstation processing units that many users are familiar with. Once that is done, message passing can be debugged using only two or four nodes or even using an off-line simulator available from Intel. Then the application can easily be scaled up to the full system. A second major advantage is the fact that from our experience it is fairly easy to obtain moderately respectable performance using the system. Almost all codes that have employed reasonably well conceived algorithms have achieved at least 100 MFLOPS on 128 nodes, and many run at significantly higher rates.

One weakness of the current system is that single node performance is disappointing, even with 40 MHz processors and the Portland Group Fortran compiler. Some Fortran codes run at 8 MFLOPS (double precision) on a single node, but many more only run at 2 to 4 MFLOPS. Since the peak performance of the i860 is 60 MFLOPS (double precision), there is ample room for improvement here. However, from our analyses [6] we do not expect more than about 8 MFLOPS per node on most Fortran codes even with the best Fortran compiler, due mainly to the somewhat limited bandwidth between main memory and the i860 processor. We have found that the 8 kilobyte on-chip data cache in the i860 is too small to significantly improve performance on most codes. A much larger cache would definitely help, but improved main memory bandwidth would be more valuable for the main body of Fortran scientific codes.

Once the Fortran compiler has been improved, the limited bandwidth of the current interprocessor network will loom as a serious performance bottleneck. Obviously Intel recognizes this problem, and the new Touchstone Delta system to be installed at CalTech will feature a grid network with much higher bandwidth. Needless to say, we will be quite interested to see what improvement in overall performance results from this upgrade.

Perhaps the most annoying drawback of the current system is the instability of the operating system. Even one year after installation, it is not unusual for the system to have

to be rebooted two or three times in a single day. Clearly these software bugs must be fixed. Related to this problem is the equally serious drawback of a weak front end system, which is basically a 386 personal computer. Fortunately, with the Sun-based Portland Group compilers, we do not have to compile and link on this system anymore. But in any event this front end is not designed to accommodate numerous interactive users, and future versions of the iPSC must include a much more powerful service facility.

Conclusions

When we compare the performance rates that we have obtained so far, the products of reasonable efforts by bright researchers, with the theoretical peak performance rates of these parallel systems, the results are somewhat disappointing: typically 1% to 5% for both the CM-2 and the iPSC/860, as compared with 30% to 60% on one processor of a Y-MP. So what do we think are the future prospects for these systems?

While we are still basically optimistic that highly parallel computers have the potential to become powerful, usable scientific systems, it is clear that some improvements must be made from the current designs. First of all, parallel vendors need to recognize that most real scientific floating point computation, and certainly scientific computation at our facility, is characterized by a ratio of floating point operations to main memory references of approximately unity. What this means is that systems cannot rely solely on large register or cache utilization ratios to obtain respectable performance rates, and that improved bandwidth between processors and main memory is essential. It also means that floating point processors must deliver good performance on computations with nonunit strides.

Secondly, it is an unpleasant fact of the state of the art in computational fluid dynamics, as well as in many other numerical applications, that the demanding problems that have the most research interest require implicit numerical solution schemes, which are inherently nonlocal in character. Also, arrays often must be accessed in each of three dimensions. These facts mean that data communication, both within a single node and between nodes, is inherently nonlocal. In some cases, different algorithms can improve data locality. But the fact still remains that for advanced scientific computers to be taken seriously, they must offer respectable performance on applications with only moderate data locality. This means, among other things, that the interprocessor communication bandwidth of current systems must be greatly increased.

Thirdly, we are now solidly convinced that for a majority of all scientific applications, and for perhaps 90% of our applications, the Fortran-90 language [4] will be the language of choice, and vendors must support this language for multiprocessor computation. Many scientists have told us that their chief reservation of porting codes to our parallel computers is the prospect that their code will no longer run on other systems. Once Fortran-90 is officially adopted, which is now expected soon, scientists will be much more willing to consider recasting their codes to employ the Fortran-90 array constructs. Already Cray, for one, has enhanced its Fortran compiler to accept some of these array constructs, and some scientists on our systems now use the Crays in their efforts to port codes to the CM-2.

It may be too much to expect that future parallel computer systems will deliver re-

spectable performance from arbitrary Fortran-90 programs. For example, it is doubtful that interconnection networks will ever have the high bandwidth and low latency required for high performance when computing along each dimension of a three dimensional array code. However, in our opinion it is reasonable to expect that a parallel computer system should deliver respectable performance on codes where a reasonable effort has been made to reduce interprocessor data traffic. For example, consider the following general paradigm for a three dimensional scientific computation:

1. Perform numerical computations, using array constructs, along the first dimension, which is mapped to be within local nodes.

2. Perform an array transposition, using the Fortran-90 RESHAPE operator, so that the second dimension is now the first dimension.

3. Perform numerical computations along the first (i.e. the second) dimension.

4. Perform an array transposition so that the original third dimension is now the first dimension.

5. Perform numerical computations along the first (i.e. the third) dimension.

6. Perform an array transposition back to the original array ordering.

One feature of the above paradigm that is common to many parallel implementations is that it features two dimensional parallelism. For a $100 \times 100 \times 100$ problem, this means 10,000 way parallelism. In our opinion, if a parallel computer system cannot deliver respectable performance on a problem of this size with two dimensional parallelism, but instead requires the programmer to also find parallelism in the third dimension, then its usability for a broad range of real scientific problems is severely limited.

We certainly do not claim that the above scheme is most efficient for all three dimensional scientific applications. Even for our computational fluid dynamics programs, other schemes are often more effective. Nonetheless, it seems clear that if a parallel computer system cannot deliver respectable performance on an application coded according to this design, then it is doubtful that it will be able to deliver respectable performance on that application no matter how it is coded. In short, we regard this as a minimum requirement of a usable parallel computer.

Although a high level of sustained performance on real scientific applications, coded with reasonable effort, is the most important consideration in a parallel computer, it is clear from our experience that other aspects of these systems also need to be improved. For example, if parallel computers are ever to be widely used in scientific computation centers, then they must be capable of handling a fairly large number of interactive users (say 50), especially during daytime hours when users are debugging and upgrading their codes.

What this means in terms of hardware is that it is essential that a parallel computer be decomposable into at least ten or twenty independent subdomains. Such a hardware design may also be an advantage in the future, when scientists tackle large multidisciplinary applications. What this means in terms of software and environment is that the parallel computer system must include one or more powerful service nodes with a full Unix operating system, including networking and batch queueing facilities. It also means that the system must also be able to dynamically repartition the computational nodes (i.e. without rebooting the system), at the same time providing reasonable security against users storing data into nodes or memory locations that they do not own.

A related issue is mass storage. While both the current CM-2 and iPSC/860 systems have moderately fast and high capacity mass storage systems, it is clearly important that future systems increase the speed and capacity of mass storage commensurate with the increase in computational power. Also, it is essential that future parallel computer systems provide a high performance network interface (such as the HiPPI interface) to allow high speed data communication to workstations and archival storage.

We recognize that the development of high performance, highly usable parallel computer systems will be a challenging task. But we feel that without major improvements in both hardware and software, there is a risk that scientists will eventually tire of struggling with these systems and will return to conventional workstations and supercomputers, and parallel systems will never be adopted for mainstream supercomputing. Thus we strongly support aggressive efforts by vendors towards these goals.

Acknowledgement

The author wishes to acknowledge helpful comments and suggestions by Eric Barszcz and Tom Lasinski of the NAS Applied Research Branch and by Rod Fatoohi, Horst Simon and Sisira Weeratunga of Computer Sciences Corp.

References

1. Bailey, D. H., Barton, J., Lasinski, T., and Simon, H., "The NAS Parallel Benchmarks", Technical Report RNR-91-002, NAS Applied Research Branch, NASA Ames Research Center, January 1991.

2. Bailey, D. H., et al., "Performance Results on the Intel Touchstone Gamma Prototype", *Proceedings of the Fifth Distributed Memory Computing Conference*, April 1990, p. 1236 – 1245.

3. Barszcz, E., "One Year with an iPSC/860", Technical Report RNR-91-001, NAS Applied Research Branch, NASA Ames Research Center, January 1991.

4. *Fortran 90*, Draft International Standard, American National Standards Institute, June 1990.

5. Hammond, S., and Barth, T. J., "An Efficient Massively Parallel Euler Solver for Unstructured Grids", RIACS Technical Report 90-47, NASA Ames Research Center, October 1990. Also published as AIAA paper 91-0441, 29th Aerospace Sciences Meeting, January 1991.

6. Lee, K., "On the Floating Point Performance of the i860 Microprocessor", Technical Report RNR-90-019, NAS Applied Research Branch, NASA Ames Research Center, October 1990.

7. Levit, C., and Jespersen, D., "Numerical Simulation of Flow Past a Tapered Cylinder", Technical Report RNR-90-021, NAS Applied Research Branch, NASA Ames Research Center, October 1990. Also published as AIAA paper 91-0751, 29th Aerospace Sciences Meeting, January 1991.

8. McDonald, J. D., "Particle Simulation in a Multiprocessor Environment", Technical Report RNR-91-003, NAS Applied Research Branch, NASA Ames Research Center, January 1991.

9. Schreiber, R., "An Assessment of the Connection Machine", RIACS Technical Report 90-47, NASA Ames Research Center, June 1990.